Differential Privacy for Growing Databases

Rachel Cummings[1], Sara Krehbiel[2], Kevin A. Lai[1], and Uthaipon (Tao) Tantipongpipat[1]
[1]Georgia Institute of Technology [2]University of Richmond

Motivation and Problem

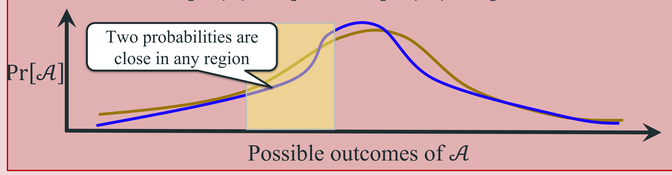
Problem: Data anonymization is **not** enough to protect user privacy
- Netflix + IMDb attack [NS2007]
- U.S. census (zip code+gender+DoB can deanonymize a person) [Swe1990]
- Hospital visit + electoral data → Medical record leak [Swe1997]

Differential Privacy
- Protects individual's privacy with mathematical guarantee
- Has been implemented by Apple, Google, Microsoft, US Census (2020), and Uber

Definition of Differential Privacy (DP)

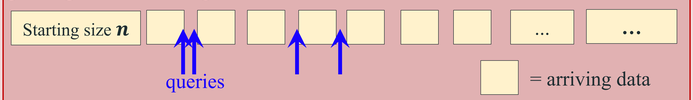
An algorithm $\mathcal{A}$ is (ϵ, δ)-DP if for any possible set of outcomes S, $\mathcal{A}$ outputs S with similar probability whether a user is included or not in the database:

$$\Pr[\mathcal{A}(D) \in S] \leq e^{\epsilon} \Pr[\mathcal{A}(D') \in S] + \delta$$



Model for Growing Databases

The large majority of DP algorithms work for *static* databases. We define the *growing* database setting as follows:



- From an initial size of n at time $t = n$, we receive 1 data point per unit of time and an arbitrary number of queries
- For query f arriving at time t, we evaluate f on the current database D_t

We say an algorithm is (α_t, β_t)-*accurate* if with probability $1 - \beta$, for each query f answered at time t, we have $|f(D_t) - a_{f,t}| \leq \alpha_t$, where $a_{f,t}$ is the algorithm's answer for query f at time t.

If $\alpha_t = \alpha$ and $\beta_t = \beta$ for all t, we call an algorithm (α, β)-*accurate*.

Related Work

- There has been minimal work on growing databases. [DNPR10] and [CSS11] give algorithms to count bits in a stream. [ST13] give algorithms to maintain private sums of real vectors arriving in a stream.
- Both of these settings correspond to only a *single* query repeatedly asked on a dynamic database. In contrast, we consider the much richer class of *linear* queries, allowing for adaptive analysis of a dynamically growing database.
- Our setting differs from *online learning* in that we desire (α_t, β_t)-accuracy (i.e., per-round accuracy guarantees) rather than regret bounds (i.e., cumulative loss bounds)

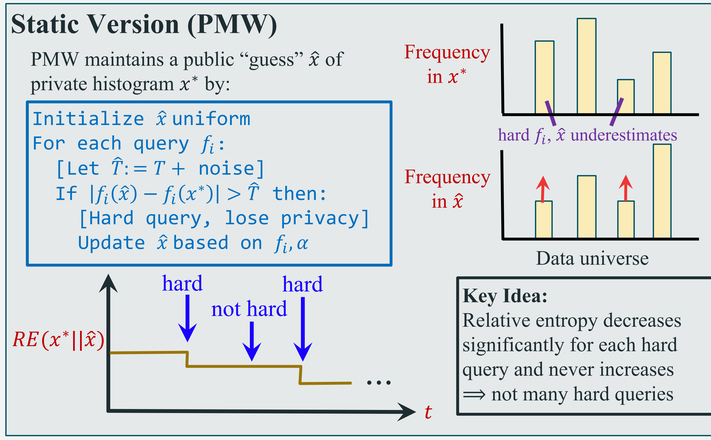
Private Multiplicative Weight Update for Growing Databases (PMWG)

Main Result:
- A novel modification PMWG that achieves the best-known accuracy bound on linear queries for growing databases
- PMWG matches the optimal PMW in the static setting up to constants
- PMWG accommodates exponential number of queries for each new data entry

Static Version (PMW)

PMW maintains a public "guess" $\hat{x}$ of private histogram x^* by:

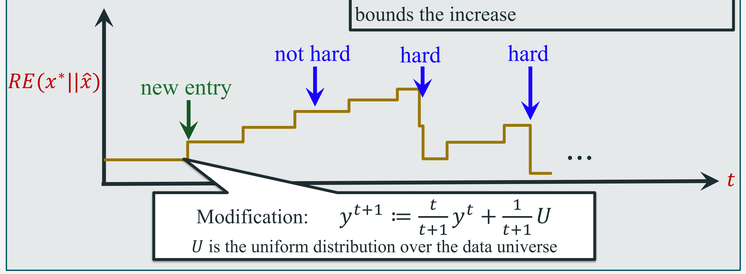
```
Initialize x̂ uniform
For each query f_i:
  [Let T̂ := T + noise]
  If |f_i(x̂) − f_i(x*)| > T̂ then:
    [Hard query, lose privacy]
    Update x̂ based on f_i, α
```



Key Idea:
Relative entropy decreases significantly for each hard query and never increases
⟹ not many hard queries

Our Algorithm (PMWG)

Key idea: Relative entropy increases when new data entries arrive, but our modification bounds the increase



Our Result (PMWG vs PMW)

PMW ([HR10])	PMWG (Our result)
$\alpha = C\left(\frac{\sqrt{\log(k/\beta)\log(1/\delta)}\log^{\frac{1}{4}} N}{\sqrt{\epsilon n}}\right)$ Answers k adaptive queries	$\alpha = C\left(\frac{\sqrt{\log(\kappa n/\beta)\log(1/\delta)}\log^{\frac{1}{4}}(Nn)}{\sqrt{\epsilon n}}\right)$ Answers $\kappa e^{c\sqrt{n}}$ more adaptive queries when each new data entry arrives

Bounds are for (ϵ, δ)-DP and (α, β)-accuracy over a database of (starting) size n and data universe of size N

Adapting Static Algorithms to the Growing Setting

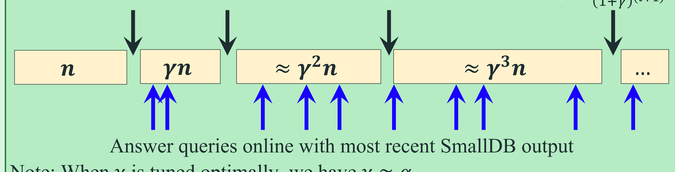
Main Result: Given a private and accurate algorithm for the static setting, we give a generic way to adapt it to the growing setting.

Fixed Accuracy Guarantees

SmallDB [BLR08] is a classic algorithm for answering a set $\mathcal{F}$ of linear queries on a static database. We illustrate our algorithm using SmallDB as an example.

Our Algorithm

Run SmallDB at times $t_i = (1+\gamma)^i n$ using privacy parameter $\epsilon_i \approx \frac{\epsilon}{(1+\gamma)^{(i+1)}}$



Answer queries online with most recent SmallDB output

Note: When γ is tuned optimally, we have $\gamma \approx \alpha$.

Static database error bound (SmallDB [BLR08])	Growing database error bound (Our result)
$\alpha = C\left(\frac{\log N \log\lvert\mathcal{F}\rvert + \log(1/\beta)}{\epsilon n}\right)^{1/3}$	$\alpha = C\left(\frac{\log N \log\lvert\mathcal{F}\rvert \log(1/\beta)}{\epsilon n}\right)^{1/5}$

Improving Accuracy Guarantees

For ERM, we want accuracy guarantees to *improve* as more data arrive. In this setting, we run the static ERM algorithm at every step and use more sophisticated tools to achieve the following accuracy guarantees under (ϵ, δ)-DP for any $p > 0$ and where d is the dimension of the data:

Static database ERM error bound ([BST14])	Growing database ERM error bound (Our result)
$\alpha_t = C\left(\frac{\sqrt{d}\log^2 t/\delta \log(1/\beta)}{\epsilon t}\right)^{1/3}$ Only applies to a single time t	$\alpha_t = C\left(\frac{d\sqrt{\log 1/\delta}\log(1/\beta)}{\sqrt{p}\epsilon t^{1/2-p}}\right)^{1/5}$ Applies to all times $t \geq n$

Extensions

- Our algorithm can transform all private and accurate static algorithms (under mild assumptions), as long as they have bounded sensitivity when new data arrives.
- Our algorithm works for static algorithms that accept adaptive or non-adaptive queries.